Data Distribution & Processing CSCI
Data Distribution CSC
Design Review

June 17, 1997
Version 1.0

1. Data Distribution & Processing CSCI

The Data Distribution & Processing CSCI is composed of the following CSCs:

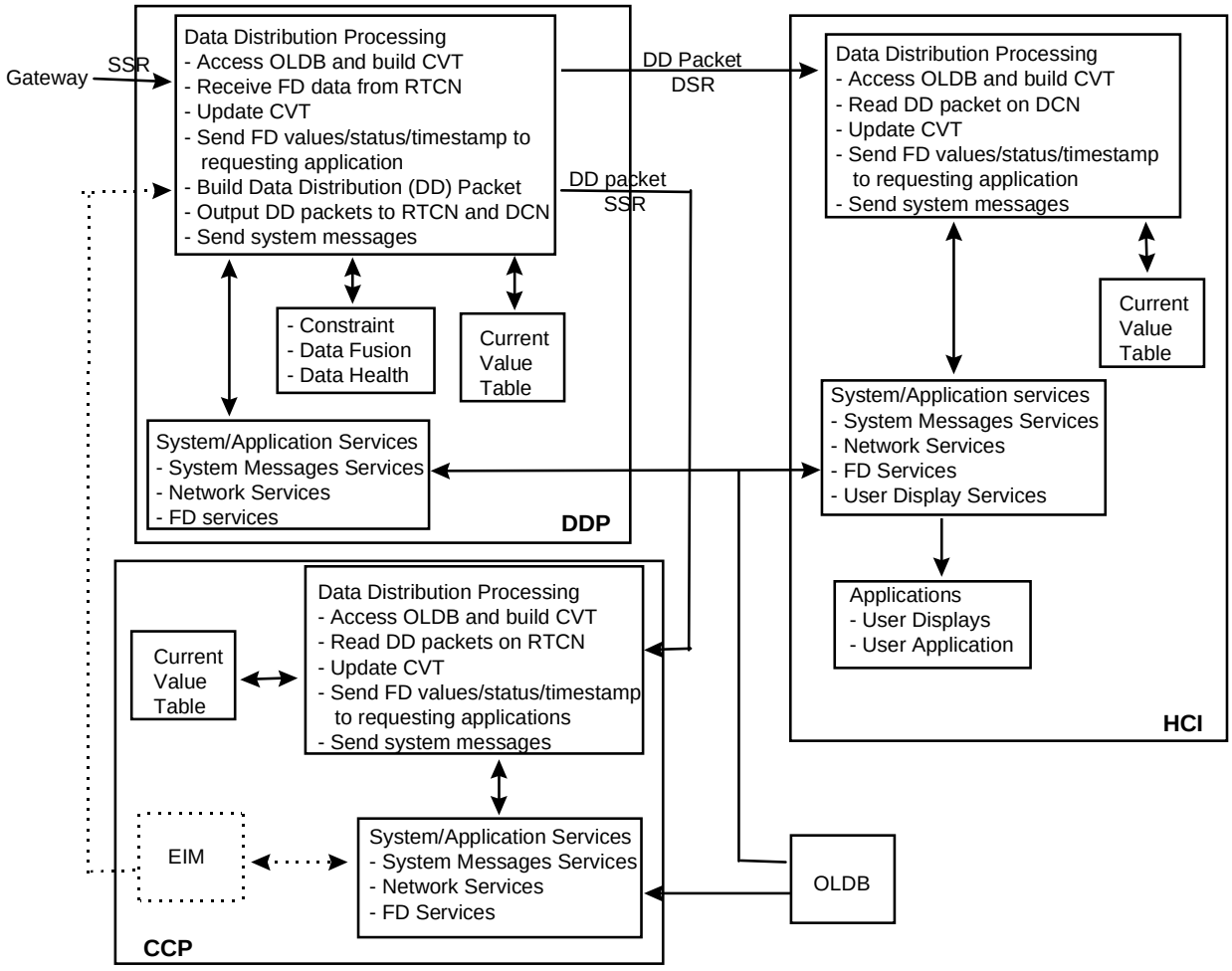
Data Distribution CSC, Data Fusion CSC, and Data Health CSC.

1.1 Data Distribution CSC Introduction

1.1.1 Data Distribution CSC Overview

The Data Distribution CSC resides in the Data Distribution Processor (DDP), the Human Computer Interface (HCI), and the Command and Control Processor (CCP). The Data Distribution CSC running in the DDP provides the capability to read FD data from the Gateways and distribute it to the RTCN and DCN. The Data Distribution CSC running in the CCP and HCI provides the capability to receive FD data from DDP and make it available to Command, user applications, and user displays.

Data Distribution CSC Overview is as follows:



1.1.2 Data Distribution CSC Operational Description

The Data Distribution CSC supports end-to-end data flow from the point where data is received at the DDP from the Gateway over the RTCN, to the point where data is delivered to the user applications and/or user displays.

1.2 Data Distribution CSC Specifications

1.2.1 Data Distribution CSC Ground Rules

1. For Redstone delivery, Data Distribution will not request a resend of data from its data source, assuming no data will be dropped via the use of Reliable Messages Capability (Resend request to Redundancy Management will be addressed in a later delivery).
2. Each Gateway stream will have an unique multicast stream name.
3. Stream names, for a given activity will be read from a file for Redstone delivery. A long term solution of including that as part of Activity definition will be addressed in a later delivery.
4. Data packet received from the same Gateway is already in ”time-order.”
5. No time validation will be performed on the packets from the Gateways.
6. Each Gateway will send data packets to the RTCN at the System Synchronous Rate (SSR).
7. Time intervals between Gateway sends for the same SSR cycle will be handled by the Gateways. Data Distribution processing will be completely data driven. DD output rate is based on input rate coming from the Gateways. Data will be output to RTCN as soon as DD processing is complete.
8. An empty packet will be sent by the Gateway if there is no data changed within an SSR cycle.
9. *Display Attributes processing for Redstone is not supported*
10. All applications, with exception of Data Fusion, Data Health, and Constraint Management, will interface with Data Distribution via FD Services.
12. Data Distribution will access the OLDB via FD Services.
13. *Application derived FDs will not be supported for Redstone.*
14. *Time Homogeneous Data Sets (THDS) will not be supported for Redstone.*
15. *The FD value override capability will not be provided for Redstone.*
16. *Persistent data/checkpoint will not be supported for Redstone.*
17. *Interface with Constraint Management will not be supported for Redstone.*
18. *Data refresh will not be supported for Redstone.*
19. *Redundancy management will not be supported for Redstone.*

1.2.2 Data Distribution CSC Functional Requirements

1.2.2.1 Data Distribution (DD) at the DDP

1. DD will receive Gateway Changed Data Packets on the RTCN.
2. DD will support at least 20 Gateways simultaneously on a single DDP.
3. DD will provide the capability to perform time-reordering on the data packets received across all the Gateways.
4. DD will provide the capability to perform time-reordering based on the time value(s) provided in the DD Packet Payload data.
5. DD will send a system message if no packet is received from a Gateway at the end of the System Synchronous Rate (SSR) cycle.
6. DD will merge data packets received from the RTCN.
7. DD will provide the capability to build Data Distribution Packets based on the CLCS DD Packet Payload format.
8. DD will distribute the following information in the Data Distribution packets:
    a. Data value
    b. Health Data status
    c. Timestamp
10. DD will provide the capability to output Data Distribution packets to the RTCN at the System Synchronous Rate (SSR).
11. DD will provide the capability to buffer Data Distribution packets and output DD packets to the DCN at the Display Synchronous Rate (DSR).

1.2.2.2 Data Distribution at the CCP

1. DD will receive Data Distribution packets on the RTCN.
2. DD will send a system message if no packet is received from the RTCN at the end of the SSR.

1.2.2.3 Data Distribution at the HCI

1. DD will receive Data Distribution packets on the DCN.
2. DD will send a system message if no packet is received from the DCN at the end of a DSR cycle.

1.2.2.4 Data Distribution (DD) at the DDP/CCP/HCI

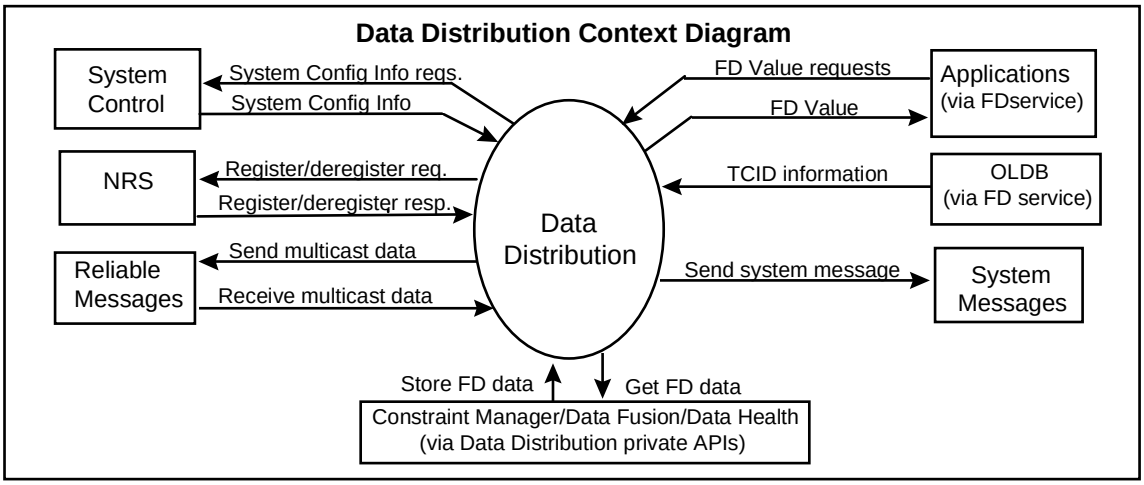
1. DD will initialize the CVT with the information contained in the Online Data Bank (OLDB).
2. DD will provide the capability to update the CVT with changed data.

3. DD will provide the capability to update the CVT with FD data:
    a. Raw or EU data
    b. Health Data Values
    c. Status/State Values
    d. Timestamp
    e. Display Attributes (TBD)
4. DD will provide an application interface allowing an application to:
    a. subscribe to FD data
    b. publish FD data
5. DD will provide an application interface allowing applications to access, on demand, the following data for an FD or a list of FDs:
    a. Raw or EU data
    b. Health Data Values
    c. Status/State Values
    d. Timestamp
6. DD will provide an application interface allowing applications to access, on a "change only" basis, the following data for a FD or a list of FDs:
    a. Raw or EU data
    b. Health Data Values
    c. Status/State Values
    d. Timestamp
7. DD will provide the capability to maintain statistics on packet rates and data rates in an internal table.

1.2.3 Data Distribution CSC Performance Requirements

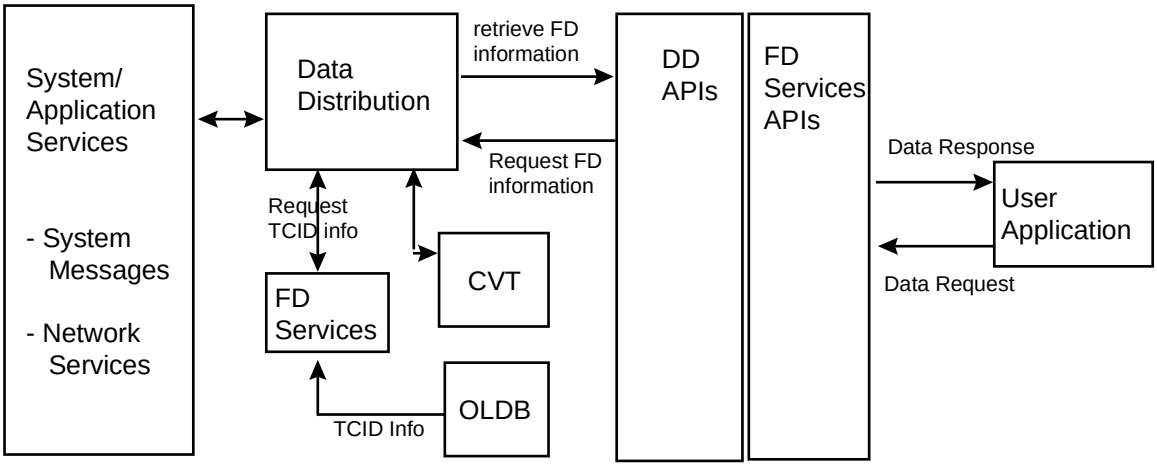
1. DD will be able to process up to 25,000 change FDs from the Gateway(s) per second.
2. DD will be able to process up to 50,000 change FDs from the Gateway(s) in a given second without losing any data.
3. DD will be able to incorporate 5,000 processed FDs (fused FDs and Health status) into the CVT per second.

1.2.4 Data Distribution CSC Interfaces

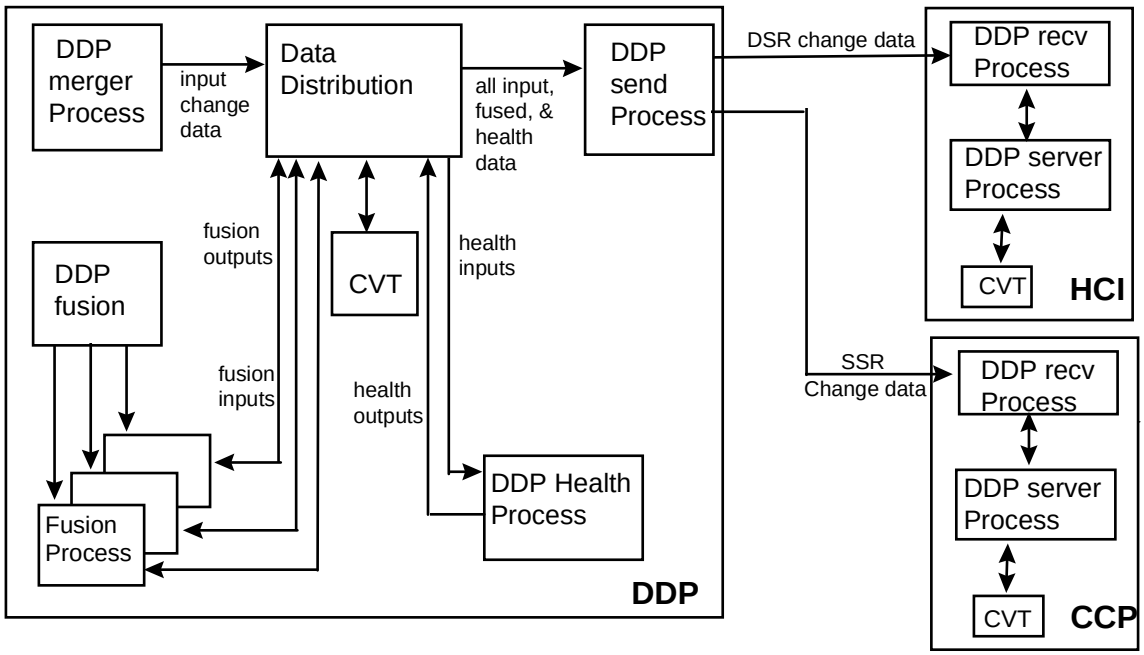


1.2.5 Data Distribution CSC Data Flow Diagram

External



Internal



1.3 Data Distribution CSC Design Specification

The Data Distribution CSC supports end-to-end data flow from the point where data is received at the DDP from the Gateways over the RTCN, to the point where data is delivered to the user applications and/or user displays.

The Data Distribution CSC on the DDP will perform these functions:

1. Receive gateway changed data packets on the RTCN
2. Perform time-reordering on the data packets received from the Gateways
3. Buffer distributed data packets and output packets to the DCN at the DSR
4. Buffer distributed data packets and output packets to the RTCN at the SSR
5. Provide data to the Data Fusion CSC and the Data Health CSC
6. Incorporate fusion and health data into the data stream

The Data Distribution CSC on the HCI will perform these functions:

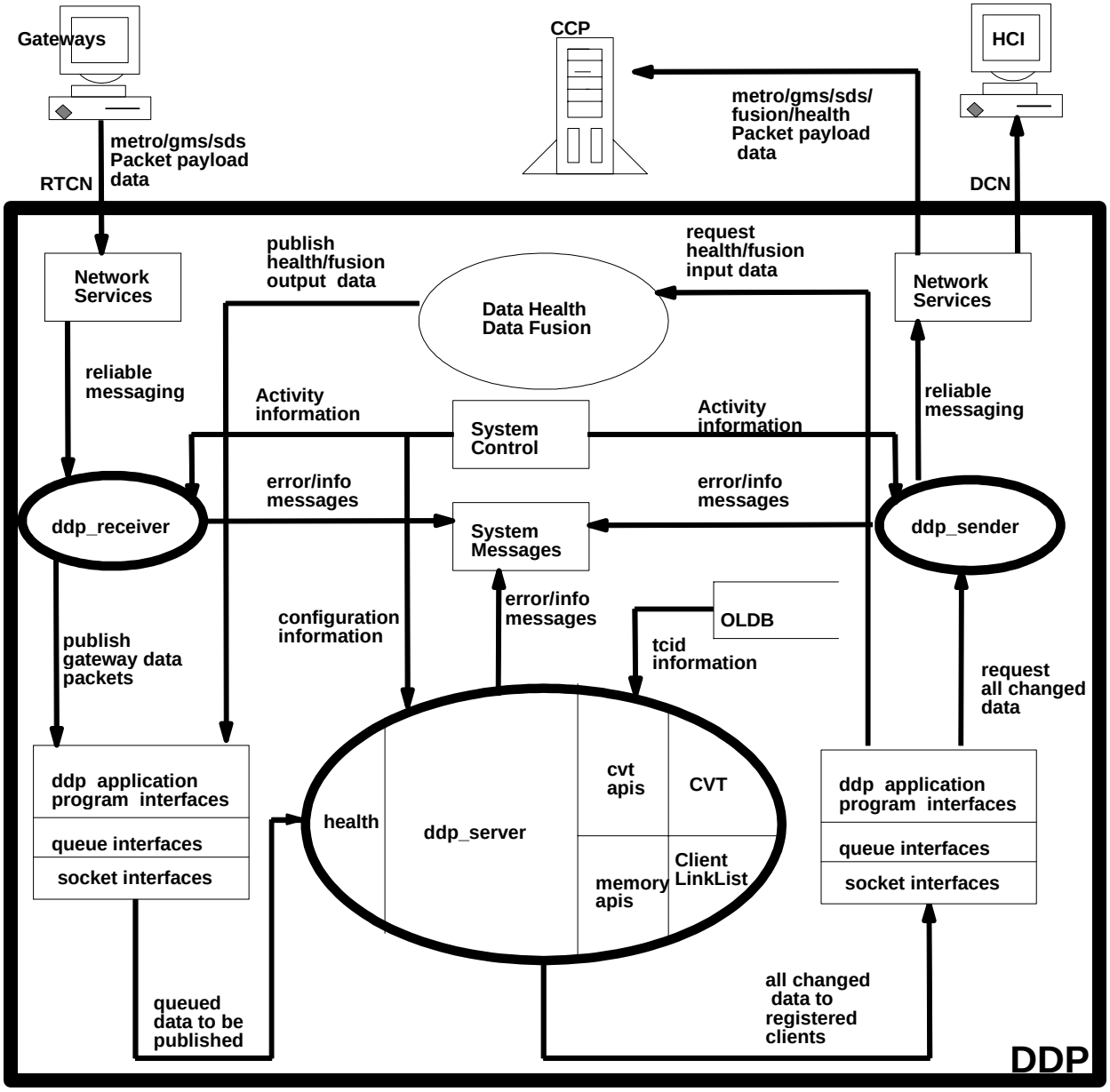
1. Receive distributed data packets on the DCN
2. Distribute packets to clients (i.e. display applications, viewers, user applications)

The Data Distribution CSC on the CCP will perform these functions

1. Receive distributed data packets on the RTCN
2. Distribute packets to clients (i.e. end item managers)

1.3.1 Data Distribution Detailed Data Flow

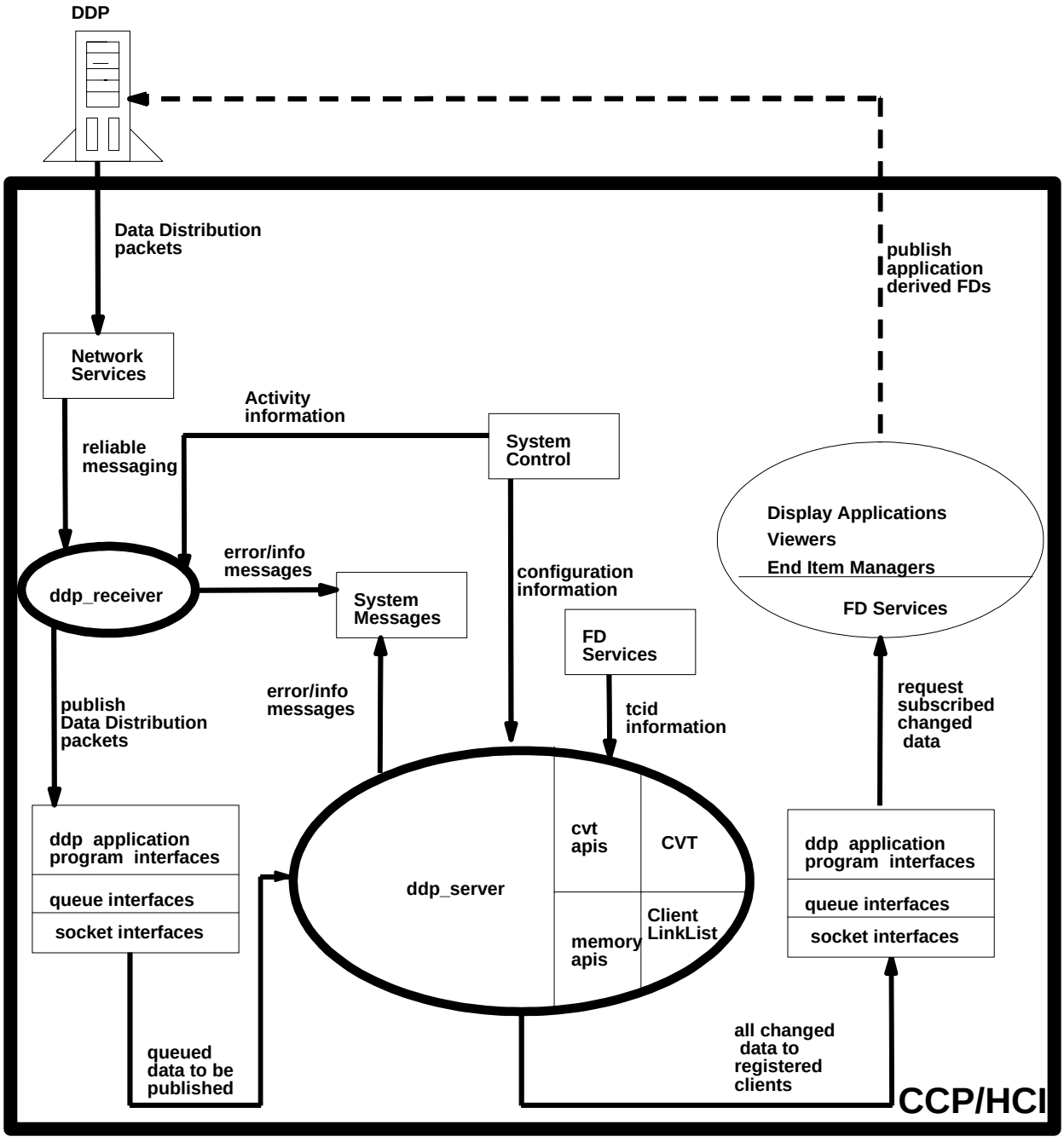
1.3.1.1 DDP Detailed Data Flow



The Data Distribution CSC interfaces with Network Services both to receive data from the Gateways, and send data out to the CCP and HCIs. Error conditions that are encountered by Data Distribution are written to System Messaging. At initialization, System Control provides the configuration information to build the multicast addresses, and determine the number of Gateways.

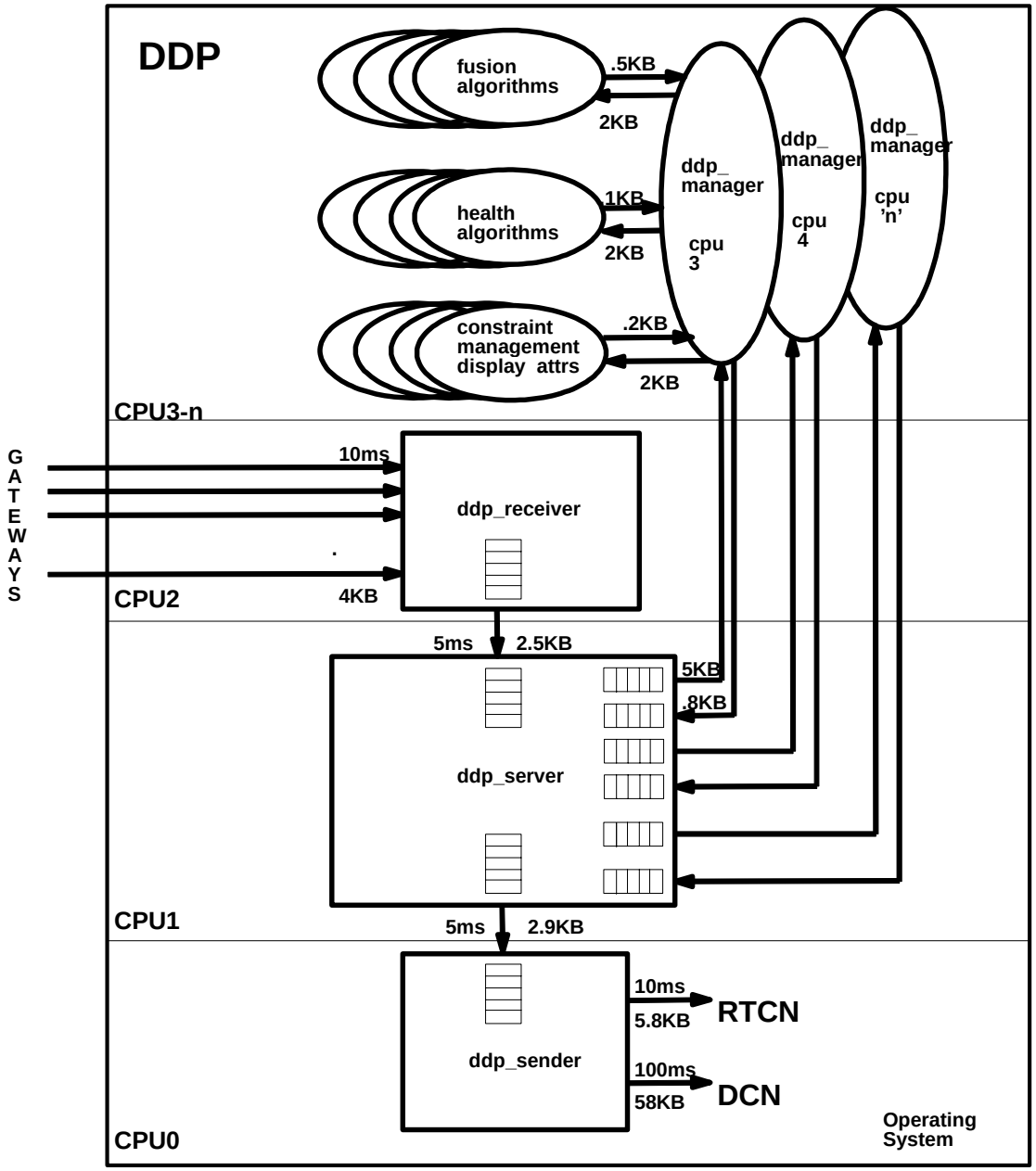
Data is merged from the Gateways in the ddp_receiver process and sent in time order to the ddp_server. The ddp_server process stores data in the current value table and sends all changed values to the ddp_sender process, the Data Fusion CSC, and the Data Health CSC. The ddp_sender process buffers the changed values into 10ms buffers and 100ms buffers to output to the CCP and HCIs respectively.

1.3.1.2 CCP/HCI Detailed Data Flow



The ddp_receiver process on the HCI and CCP receives data from the DDP at 10ms (RTCN SSR) and 100ms (DCN DSR) rates on the CCP and HCI's respectively. The ddp_receiver parses the data buffer and publishes the changed data values to the ddp_server. The ddp_server process stores the changed values in the CVT. The ddp_server sends the changed values to the clients (i.e. display applications, viewers, and end item managers).

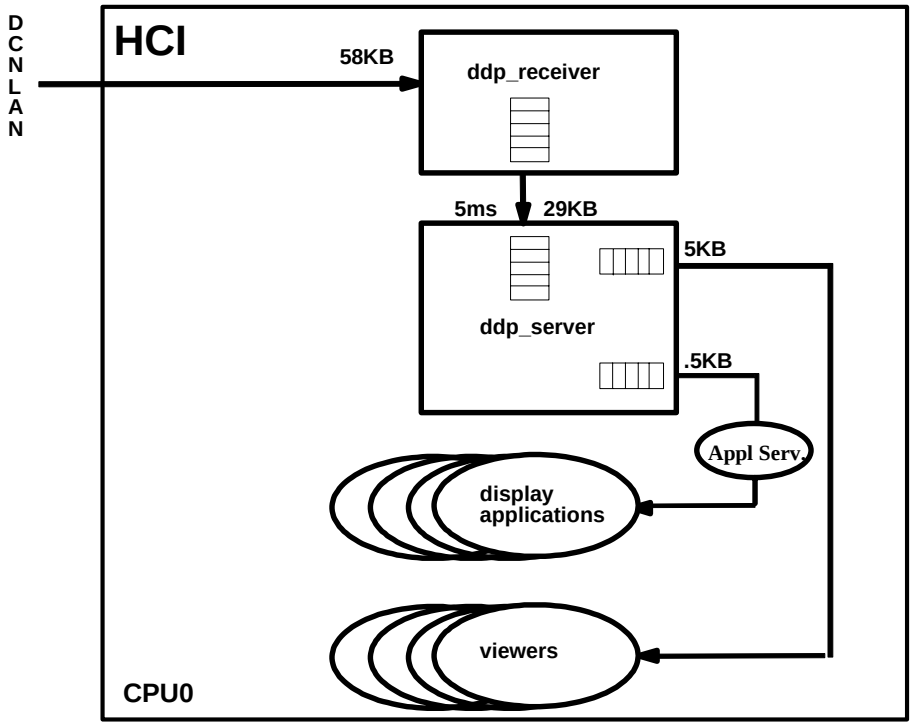
1.3.1.3 Data Distribution Proposed Performance Layout



Data Flow Parameters

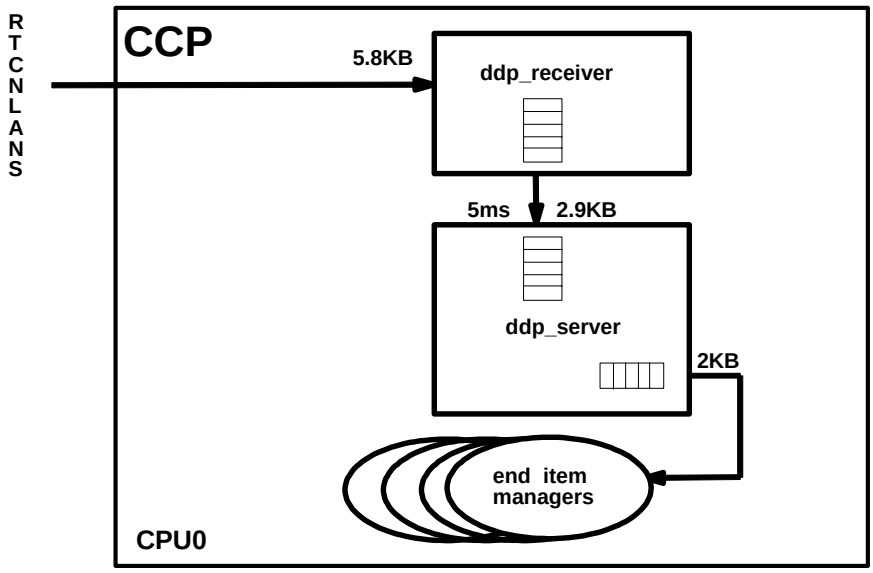
1. Approximately 40K FDs in the OLDB (FDs from gateway only; 80K FDs with fusion; ~100K FDs with pseudo FDs).
2. Average of 8 bytes per packet (32 bit analog).
3. 4KB bytes per 10ms from the Gateway (8bytes * 500 samples = 4K per ms)
4. Approximately 5KB transferred from the receiver every 10ms (4K + hdr/time delta)

5. Approximately 5.8KB transferred from the DDP every 10ms (Gateway data and fusion/health data)
6. Approximately 20ms latency (10ms in the receiver collecting Gateway data, and 10ms in the server/sender)



Data Flow Parameters

1. Approximately 58K will be received every 100ms from the DDP at the HCI
2. Approximately 5KB transferred from the server to the clients every 100ms



Data Flow Parameters

1. Approximately 5.8K will be received every 10ms from the DDP at the CCP
2. Approximately 2KB transferred from the server to the clients every 10ms

1.3.2 Data Distribution External Interfaces

1.3.2.1 Data Distribution Message Formats

1.3.2.1.1 ddp_receiver messages

Message Number = 1
Message Group = DDP
Severity = Informational

DDP Receiver is initialized

Help Information Content:
The receiver process has initialized successfully

Detailed Information:
n/a

Message Number = 2
Message Group = DDP
Severity = Critical

DDP Receiver was unable to establish a connection with the server, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
During initialization of the DDP services, there was a problem creating a socket connection between the ddp_server and the ddp_receiver.

Detailed Information:
n/a

Message Number = 3
Message Group = DDP
Severity = Major

DDP Receiver was unable to send an event to the ddp_server, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_receiver was unable to send the data packet event to the ddp_server, so there must be a problem with the socket connection.

Detailed Information:
Make sure the ddp_server process is still active.
Check to see if the queue is full.

Message Number = 4
Message Group = DDP
Severity = Critical

DDP Receiver was unable to retrieve the configuration information, system_error = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing System Control error number value
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_receiver was unable to retrieve the configuration information needed to define the multicast address and determine the number of Gateways to be read.

Detailed Information:
For Redstone:
Make sure the configuration file exists.
Make sure the file has read access.

For Thor:
See System Control for error conditions.

Message Number = 5
Message Group = DDP
Severity = Critical

DDP Receiver was unable to establish a connection with the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing Network Services error number value

Help Information Content:
The ddp_receiver was unable to open the lan connection via ”clm_open” from Network Services.

Detailed Information:
Refer to error conditions provided by Network Services.

Message Number = 6
Message Group = DDP
Severity = Informational

DDP Receiver established connection with the LAN, mcaddr = #ARGUMENT1#

ARGUMENT1 = multicast address

Help Information Content:
n/a

Detailed Information:
n/a

Message Number = 7
Message Group = DDP
Severity = Major

DDP Receiver lost the connection with the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing Network Services error number value

Help Information Content:
The ddp_receiver was unable to receive data via "clm_recv" from Network Services.

Detailed Information:
Refer to error conditions provided by Network Services.

Message Number = 8
Message Group = DDP
Severity = Informational

DDP Receiver detected that gateway #ARGUMENT1# has become inactive

ARGUMENT1 = Multicast address for gateway that went inactive

Help Information Content:
The ddp_receiver was unable to receive data from a gateway after TBD milliseconds.

Detailed Information:
Check to make sure the gateway is sending out data.

Message Number = 9
Message Group = DDP
Severity = Informational

DDP Receiver detected that gateway #ARGUMENT1# has become active

ARGUMENT1 = Multicast address for gateway that became active

Help Information Content:
The ddp_receiver is received data after it was previously inactive.

Detailed Information:
n/a

Message Number = 10
Message Group = DDP
Severity = Informational

DDP Receiver is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.1.2 ddp_server messages

Message Number = 11
Message Group = DDP
Severity = Informational

DDP Server is initialized

Help Information Content:
The server process has initialized successfully.

Detailed Information:
n/a

Message Number = 12
Message Group = DDP
Severity = Critical

DDP Server detected an error reading from the OLDB, oldb_error = #ARGUMENT1#, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing the FD services error number value
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_server was unable to read the online data bank information for initializing the CVT.

Detailed Information:
See fd_services error numbers for information.

Message Number = 13
Message Group = DDP
Severity = Critical

DDP Server detected an error creating the CVT, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_server was unable to allocate shared memory to create the current value table.

Detailed Information:
Use the UNIX errno to determine the problem.

Message Number = 14
Message Group = DDP
Severity = Critical (on DDP), Major/Minor, based on client on (CCP/HCI)

DDP Server lost the connection with a client, name = #ARGUMENT1#, pid = #ARGUMENT2#

ARGUMENT1 = Process name of client
ARGUMENT2 = Process id of client

Help Information Content:
The ddp_server lost the socket connection with the client. The client will detect it and try to reconnect.

Detailed Information:
n/a

Message Number = 15
Message Group = DDP
Severity = Informational

DDP Server is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.1.3 ddp_sender messages

Message Number = 16
Message Group = DDP
Severity = Informational

DDP Sender is initialized

Help Information Content:
The sender process has initialized successfully.

Detailed Information:
n/a

Message Number = 17
Message Group = DDP
Severity = Critical

DDP Sender was unable to establish a connection with the server, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
During initialization of the DDP services, there was a problem creating a socket connection between the ddp_server and the ddp_sender.

Detailed Information:
n/a

Message Number = 18
Message Group = DDP
Severity = Critical

DDP Sender was unable to retrieve the configuration information, system_error = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing System Control error number value
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_receiver was unable to retrieve the configuration information needed to define the multicast address and determine the number of Gateways to be read.

Detailed Information:
For Redstone:
Make sure the configuration file exists.
Make sure the file has read access.

For Thor:
See System Control for error conditions.

Message Number = 19
Message Group = DDP
Severity = Informational

DDP Sender established connection with the LAN, mcaddr = #ARGUMENT1#

ARGUMENT1 = multicast address

Help Information Content:
n/a

Detailed Information:
n/a

Message Number = 20
Message Group = DDP
Severity = Critical

DDP Sender lost the connection with the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing Network Services error number value

Help Information Content:
The ddp_sender was unable to open the lan connection via ”clm_open” from Network Services.

Detailed Information:

Refer to error conditions provided by Network Services.

Message Number = 21
Message Group = DDP
Severity = Major

DDP Sender was unable to write to the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT1#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_sender was unable to write to the multicast address.

Detailed Information:
Make sure the multicast address is still open.

Message Number = 22
Message Group = DDP
Severity = Informational

DDP Sender is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.2 Data Distribution Display Formats

There are no display formats for the DD CSC.

1.3.2.3 Data Distribution Input Formats

There are no input formats for the DD CSC.

1.3.2.4 Data Distribution Recorded Data

Statistical data will be recorded to a file. TBD

1.3.2.5 Data Distribution Printer Formats

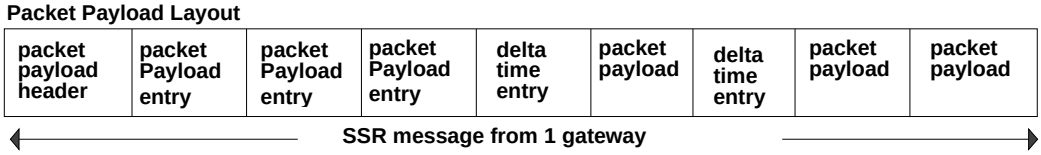
There are no printer formats for the DD CSC.

1.3.2.6 Data Distribution Inter-process Communication

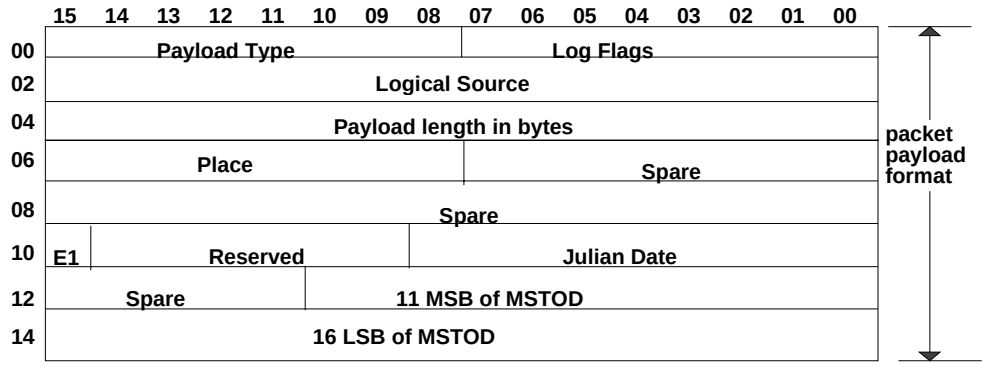
1.3.2.6.1 Network Services - Packet Payload Formats

These are the packet formats received from the Gateway machines (Ref. RTPS Packet Payload ICD, 84K00351.000, June 2, 1997, Pre-Release 1 for details).

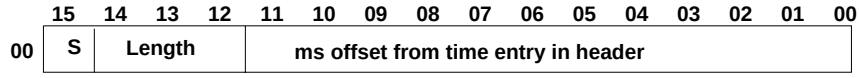
Packet Payload Layout



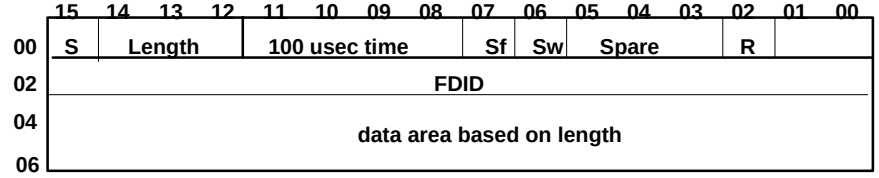
Packet Payload Header



Delta Time Entry



Packet Payload Entry



1.3.2.6.2 Distributed Data Packet Messages

This is the packet format that is distributed to the applications for data requests from the ddp_server process.

Distributed Packet Layout

Packet Payload Header

<see Packet Payload Header in section 1.3.2.6.1>

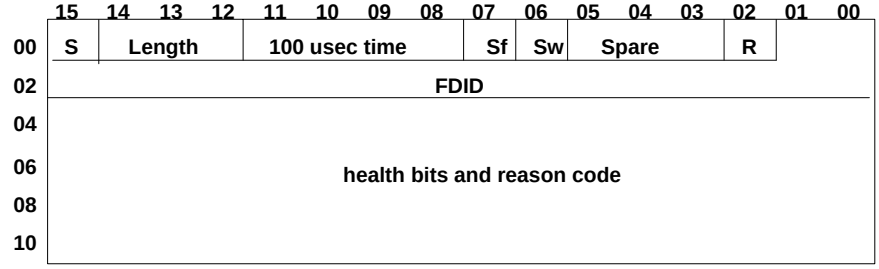
Delta Time Entry

<see Payload Time Entry in section 1.3.2.6.1>

Packet Payload Entry

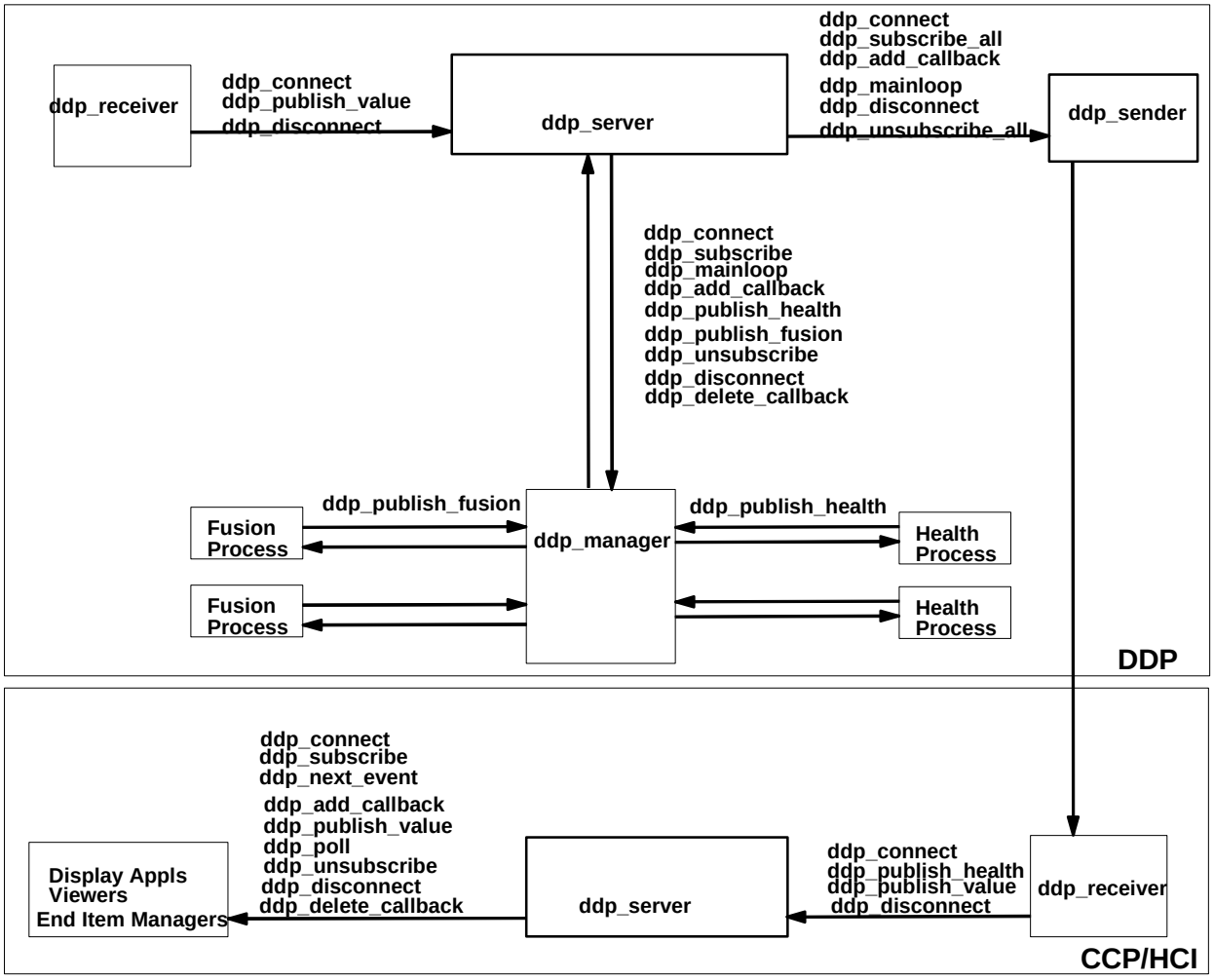
<see Packet Payload Entry in section 1.2.3.6.1>

Packet Health Entry



1.3.2.7 Data Distribution External Interface Calls

1.3.2.7.1 Current Value Table Interfaces



Initialization/startup:

ddp_app_add_input

Add socket input to be monitored by Xtmainloop.

ddp_add_callback

Add an event callback function to be invoked when changes arrive.

ddp_connect

Establishes a client connection with the ddp_server.

ddp_get_client_socket

Retrieves the client socket identifier.

ddp_read_socket

Retrieves information from the client socket.

ddp_reconnect

Performs a reconnection to the server if the connection was broken.

ddp_signals

Sets a default set of signal handlers.

ddp_subscribe

Tells the server to notify client when specified FD health/value changes.

ddp_subscribe_all

Tells the server to notify client when any FD health/value changes.

ddp_flush

Flush the clients event queue.

Operational:

ddp_app_remove_input

Delete the socket connection added from the ddp_app_add_input.

ddp_dispatch_event

Reads the socket and invokes the callbacks.

ddp_mainloop

Indefinitely waits for incoming events and invokes callbacks.

ddp_next_event

Performs a ”select” waiting for incoming events.

ddp_poll

Request the health/value for a specified FD.

ddp_publish_fusion

Notifies the ddp_server to store the fusion result in the CVT for an FD.

ddp_publish_health

Notifies the ddp_server to store the health bits and reason code in the CVT for an FD.

ddp_publish_value

Notifies the ddp_server to store the value in the CVT for an FD.

Termination:

ddp_delete_callback

Removes the callback from the event notification.

ddp_disable_fds

Notifies the server to stop sending events for changed FDs.

ddp_disconnect

Performs disconnection from the server and close the socket connection.

ddp_unsubscribe

Notifies the ddp_server to unsubscribe to specified FDs.

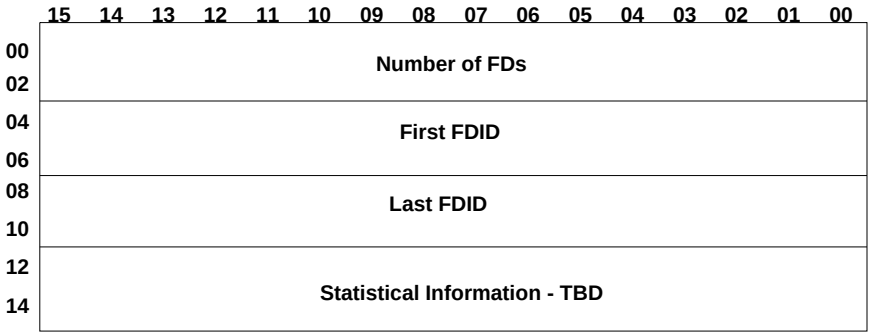
ddp_unsubscribe_all

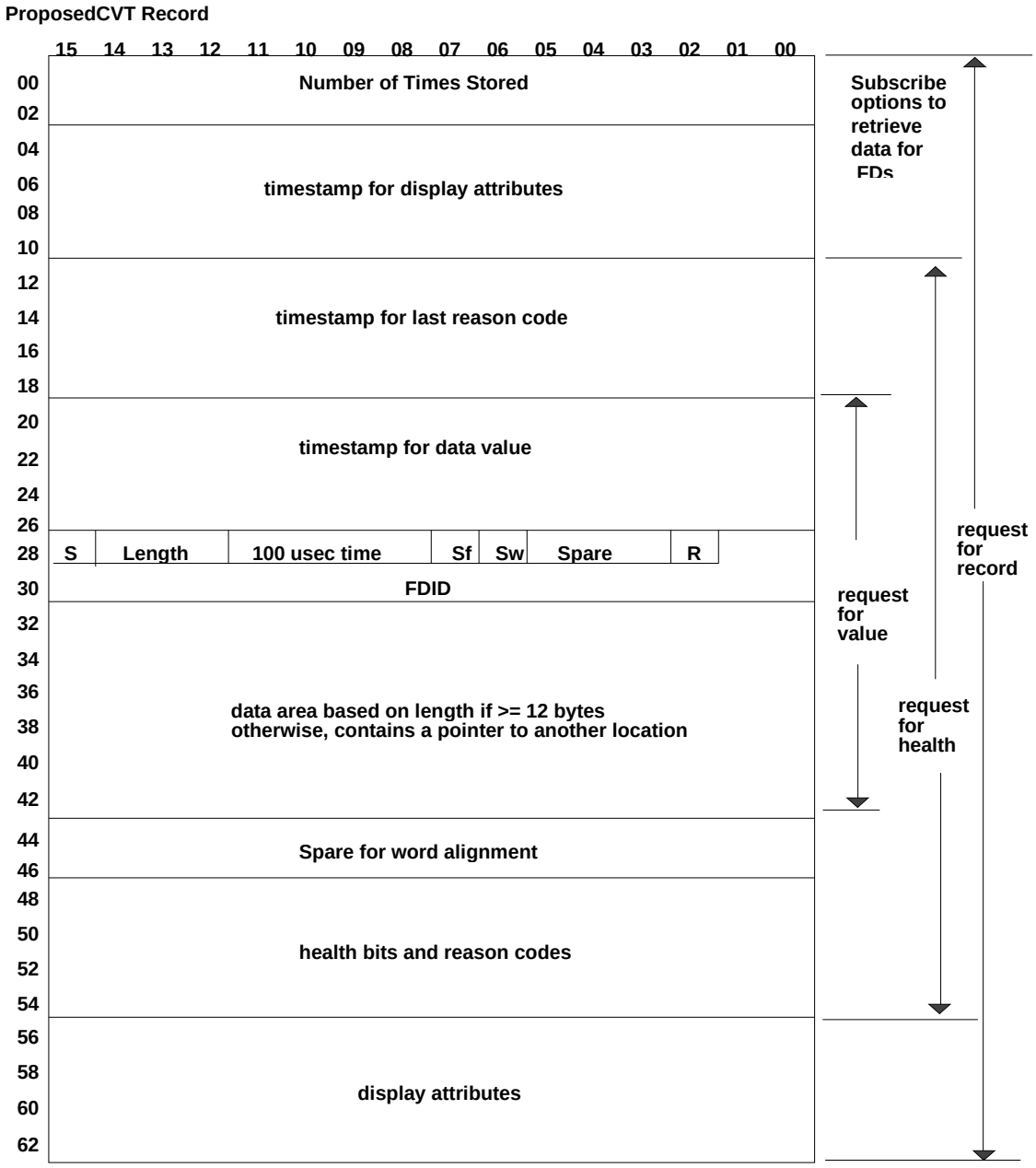
Notifies the ddp_server to unsubscribe to all FDs.

1.3.3 Data Distribution Internal Interfaces

1.3.3.1 Current Value Table Format

CVT Header

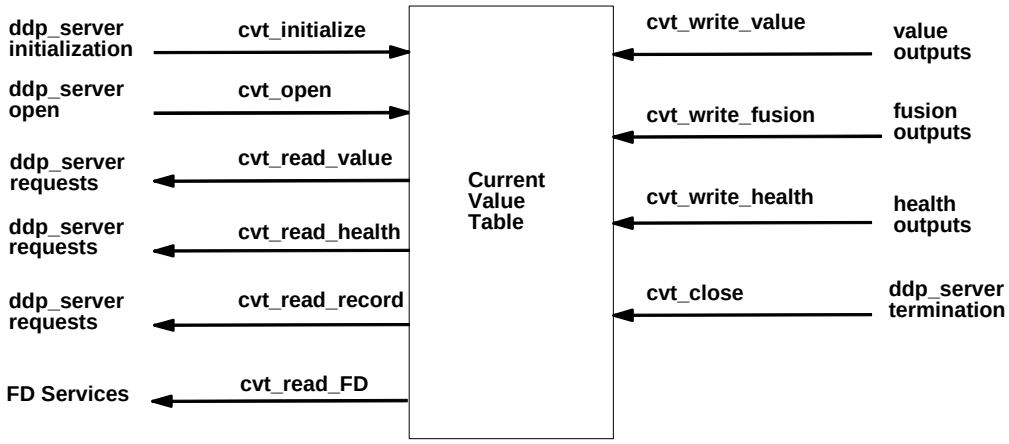




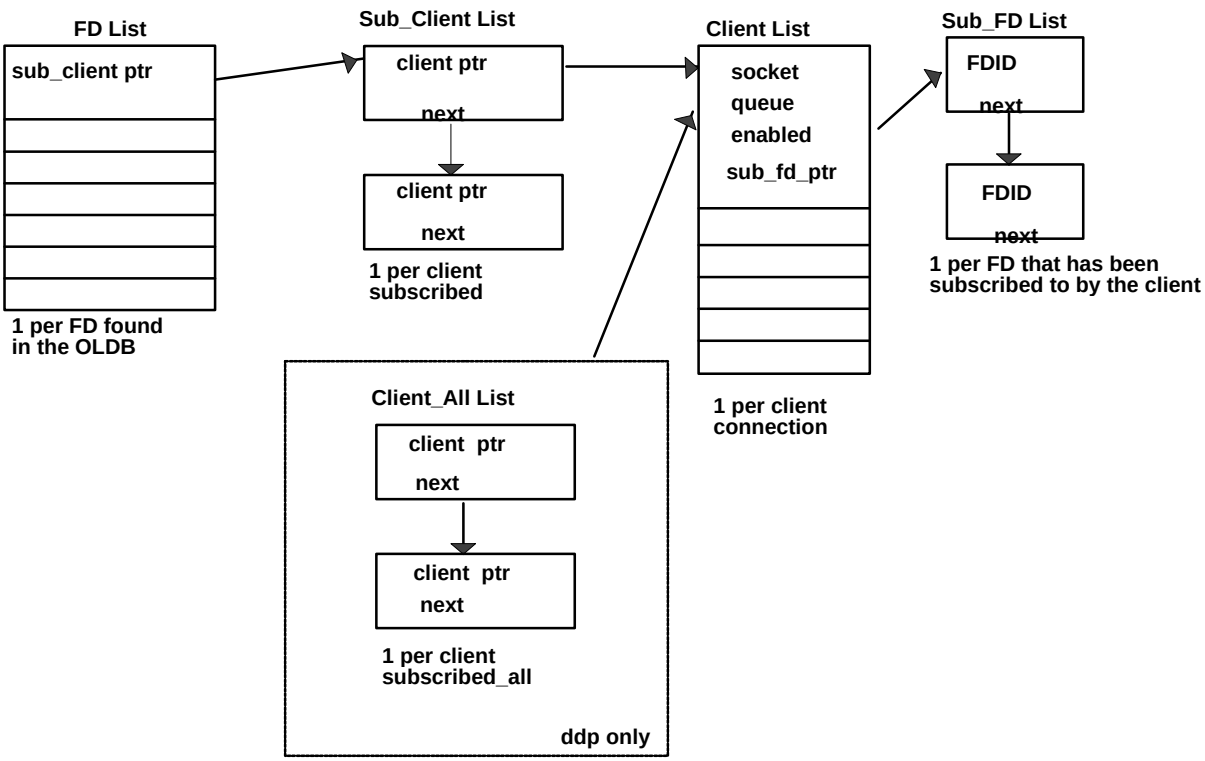
The Current Value Table (CVT) contains the latest value, health, and display attributes for each valid FD. The CVT uses shared memory to enable a restrictive set of clients (i.e. performance monitoring, debuggers), to view the contents without impacting the performance of the main processing.

The CVT is an array indexed by FD identifier (FDID). The CVT is initialized with values of certain fields from the OLDB for fast access by user applications (i.e. engineering units). The Payload Packet is stored in its entirety for performance reasons. A timestamp for the last changed value, a timestamp for the last changed health bits, error/status code, timestamp for display attributes, and display attributes are also stored in the CVT.

The following CVT interfaces are provided for maintaining the CVT. They are used by Data Distribution and FD Services for direct access into CVT. The CVT interfaces are shown below:



1.3.3.2 Client List Table Formats



Clients on the DDP are the ddp_receiver, ddp_sender, and the ddp_manager.
Clients on the HCI are the display applications, viewers, and user applications.
Clients on the CCP are the end item managers.

When clients connect to the ddp_server, an entry is added to the CLIENT LIST. The entry contains socket information for notifying the client if values change.

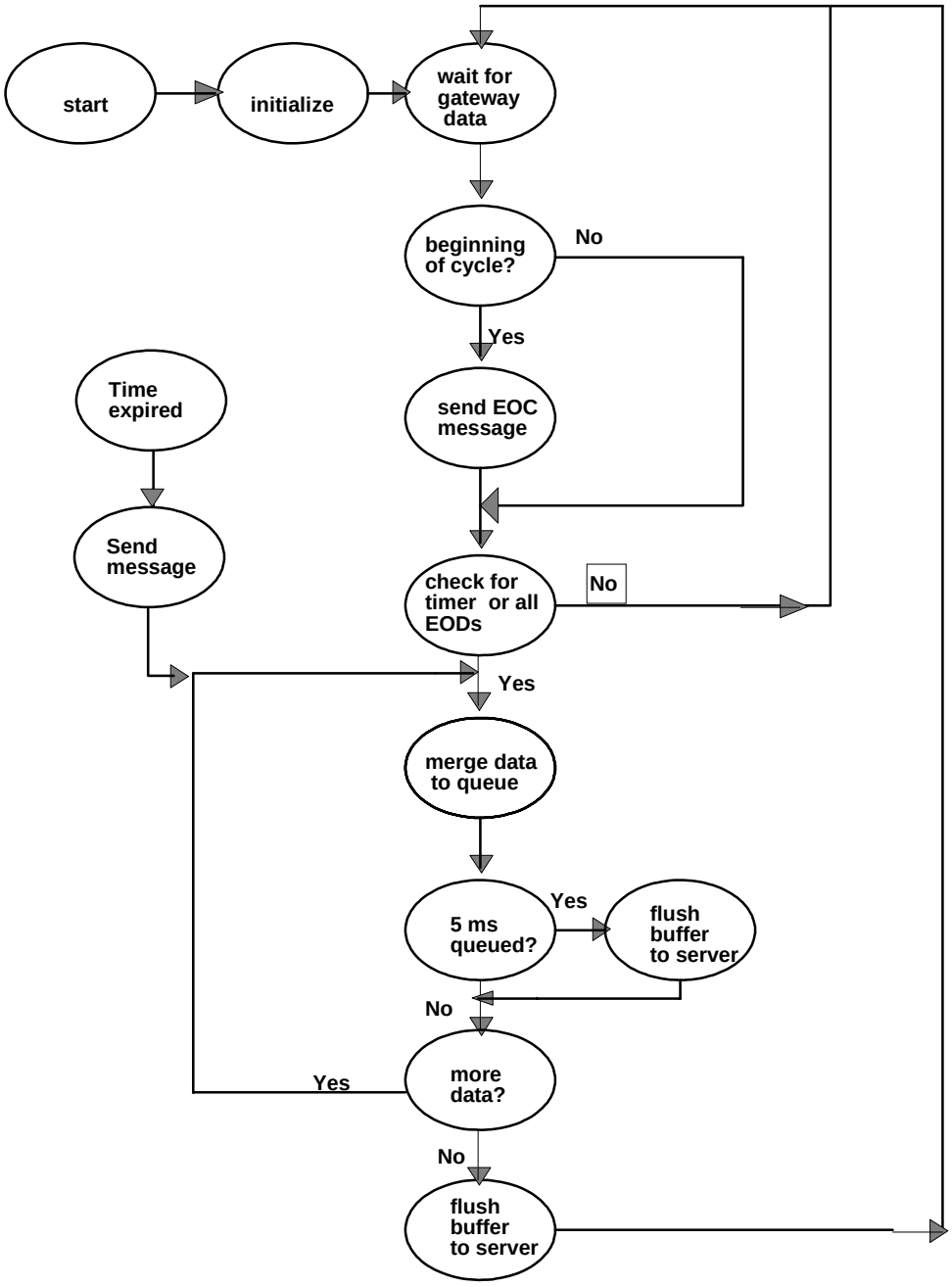
When clients subscribe to FDs, the SUB_FD list is defined with one entry for every request.

The FD_LIST is initialized containing an entry for every FD. It contains a pointer to a linked list for all clients subscribed to that FD. When an FD is published, the SUB Client LIST is searched and an event is placed on the queue for every subscribed client.

Each time an FD changes, the CLIENT_ALL list is searched and an event is placed on the queue for every client

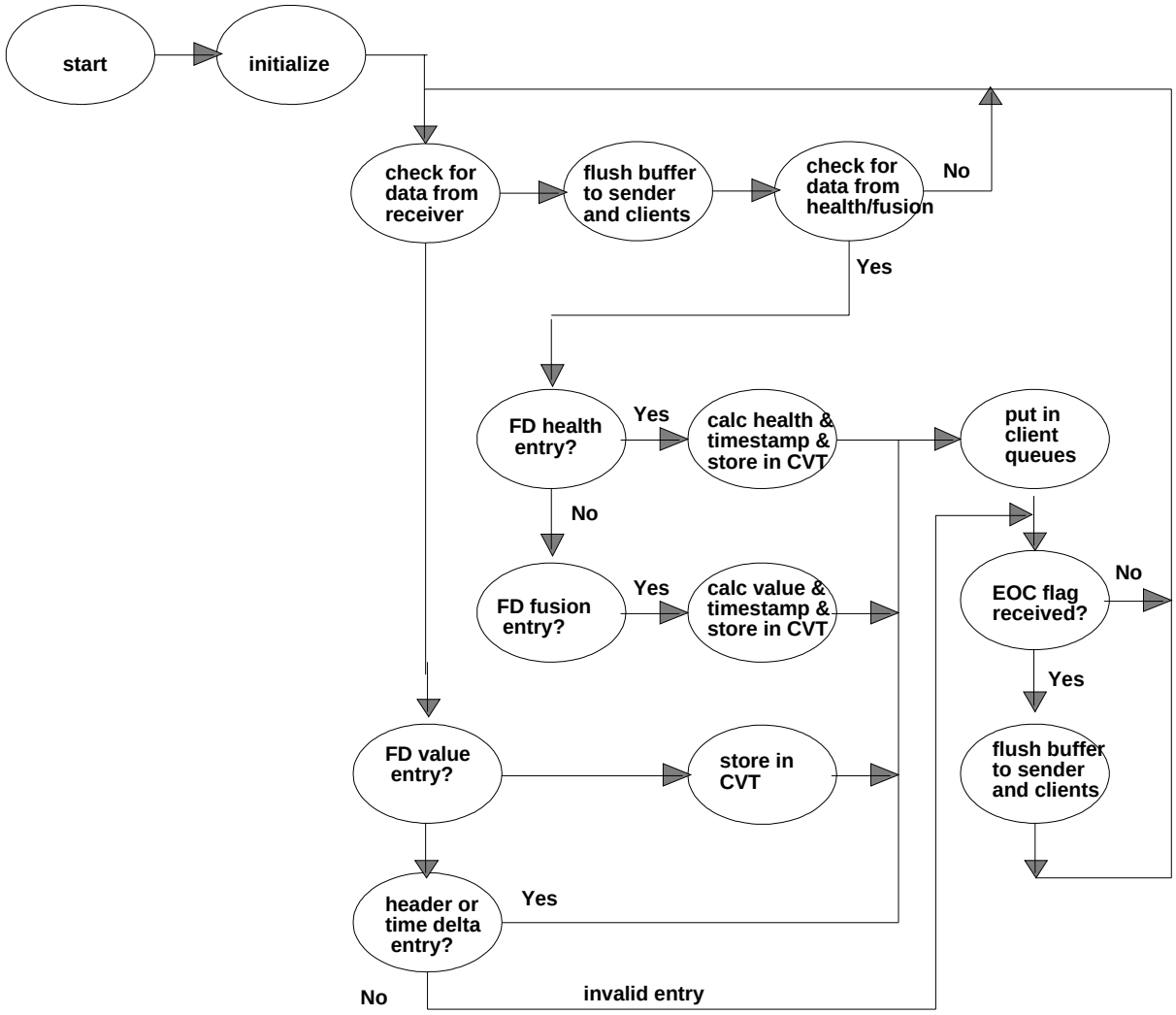
1.3.4 Data Distribution Structure Diagram

1.3.4.1 Receiver Object



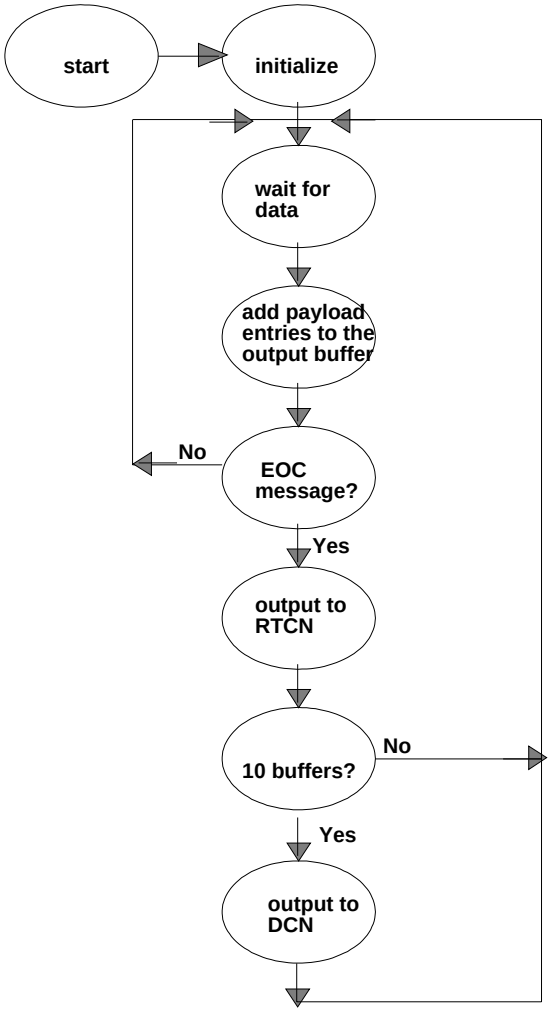
1. The 1st Header (1every 10ms) is passed to the to ddp_server.
2. Time delta packets (max of 9 every 10ms) are passed to the ddp_server as needed.
3. An EOC message is passed to the ddp_server at end of 10ms.
4. An EOC message is passed if no data is available in 10ms time period.
5. The buffer queue is flushed for every 5ms worth of data.

1.3.4.1.1 Server Object



1. The server process receives data from the receiver process, fusion/health CSCs.
2. The server will only process data from the fusion and health CSCs if there is no data to be processed from the ddp_receiver.
3. The packet being processed will be timestamped (if from fusion or health), and stored in the CVT, unless it is a header, time delta, or EOC message.
4. The packet will be placed in the ddp_sender and fusion/health CSCs queues.
5. Once the EOC message is processed, the EOC message is placed on the output queue and the queues are flushed.

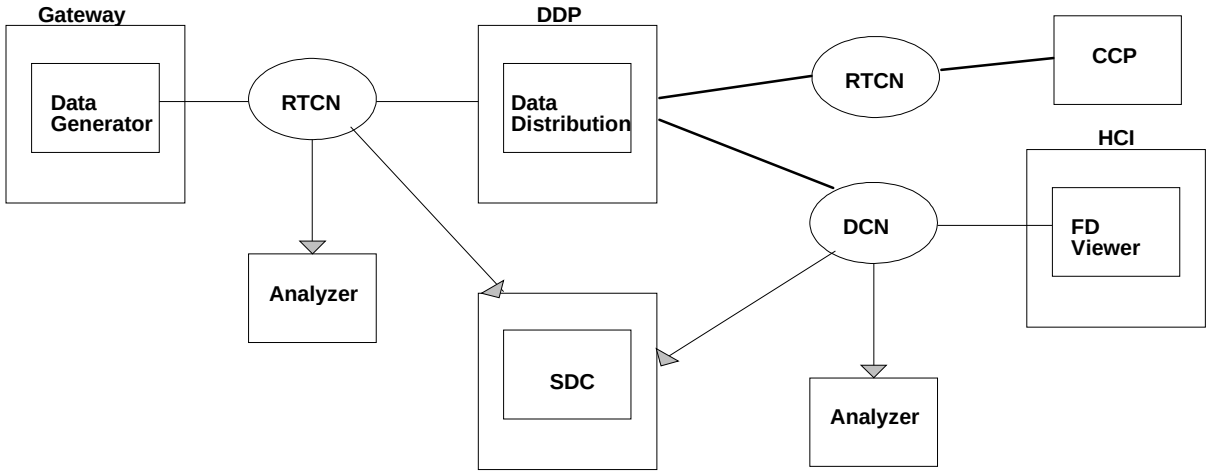
1.3.4.1.2 Sender Object



1. The sender process will receive packets from the ddp_server.
2. When the EOC message is received, the sender will output the buffer to the RTCN LAN using reliable messaging.
3. If 10 packets have been sent to the RTCN, the sender will output the composite buffer to the DCN LAN using reliable messaging.

1.3.5 Data Distribution Test Plan

1.3.5.1 Environment



1.3.5.2 Test Tools

1. Data Generator - used to output up to 8 data streams to the DDP.
2. PC Goal - used to compare data values with values on the HCI.
3. SDC - used to validate data offline.

1.3.5.2.1 Test Cases

1.3.5.3 Receiver

1. Verify ddp_receive can establish a connection with ddp_server.
2. Verify ddp_receive can establish a connection with LAN.
3. Verify ddp_receive can receive multiple data streams from the LAN.
4. Verify ddp_receive can receive packets at the SSR.
5. Verfiy ddp_receive can time-order FD change packets.
6. Verify ddp_receive can publish to ddp_server.
7. Verify ddp_receive can realize when a gateway becomes nonactive/active.

1.3.5.3.1 Server

1. Verify the ddp_server can accept client connections.
2. Verify the ddp_server can send FD data to a polling client.
3. Verify the ddp_server can send FD data to a client after every change.
4. Verify health bits are maintained if FDs are changed from the receiver.
5. Verify timestamp information is added to fusion FDs.
6. Verify timestamp information is added to FDs undergoing a health change.
7. Verify an EOC message is passed through the ddp_server to all clients.
8. Verify the ddp_server initililzes correctly.
   -Read TCID (OLDB) file.

1.3.5.3.2 Sender

1. Verify that ddp_sender can establish connection with ddp_server.
2. Verify that ddp_sender can establish connection with RTCN and DCN LAN.
3. Verify that ddp_sender can build 10 ms buffer to the RTCN LAN successfully.
4. Verify that ddp_sender can build 100 ms buffer to the DCN LAN successfully.

1.3.5.3.3 Data Distribution APIs

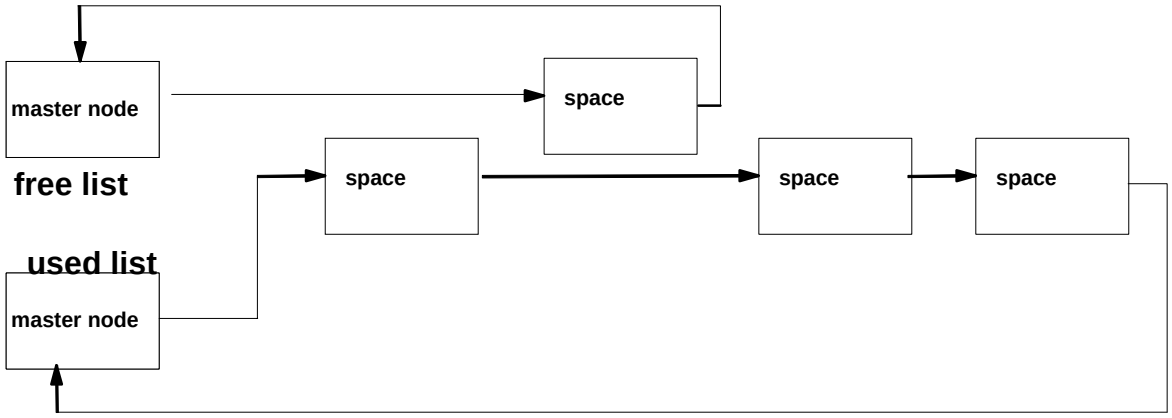
DD API testing will be accomplished by the development of simple code sequences that will exercise all the API calls for initialization/startup, operational, and termination (listed in 1.3.2.7.1).

Appendix A

Memory Private Library Object

Since mallocs are very expensive, and several small blocks of space are needed to maintain the linked-list for the client list in the ddp_server process, a memory management object will be defined to malloc large chunks of memory, and assign small blocks on an as needed basis. The space will be malloced during the ddp_space_initialize call. All space will be assigned to the free list. As space is needed, blocks will be ”moved” from the free list to the used list (which is really just pointer manipulation).

FYI: malloc() doesnt allocate space less than 32 bytes, and for each malloc() call, 8 bytes extra are allocated for internal pointers to the malloced space. A known problem is that after several mallocs for linked lists, a process becomes fragmented and swapping occurs.



- ddp_space_initialize — initialize the used and free lists
- ddp_space_get — get space from the free list
- ddp_space_put — put space from from the used list

Queue Private Library Object

Queues are used to buffer communication between the ddp_server and client applications. Queues are defined to be allocated memory where changed packets are stored in sequence to be sent to requesting processes. The queues minimize the number of system calls (which are expensive), and allow for the elasticity between the processes. The flush rate for the processes varies, based on the latency.....

• ddp_queue_create	create a queue
• ddp_queue_clear	clear a queue contents
• ddp_queue_destroy	destroy a queue
• ddp_queue_get_bufsize	retrieve queue allocated size
• ddp_queue_get_size	retrieve # bytes in queue
• ddp_queue_get_free_space	retrieve # available bytes in queue
• ddp_get_enqueue_buf	get write buffer from queue
• ddp_queue_alloc_bytes	allocate the specified bytes
• ddp_get_dequeue_buf	get read buffer from queue
• ddp_get_free_bytes	deallocate specified amount
• ddp_queue_peek_at_data	copy data into buffer
• ddp_dequeue_data	read data from queue
• ddp_enqueue_data	copy data into queue
• ddp_queue_resize	rsize queue while preserving contents
• ddp_queue_cache_grow	grow the queue cache to add fd
• ddp_queue_reset_fd	reset queue for file desc. and I/O dir
• ddp_queue_from_fd	find queue from file desc. and I/o dir
• ddp_queue_cache_flush	flush output queue to their sockets
• ddp_queue_count_cycle_bytes	raise data cycle size by byte count
• ddp_queue_get_cycle_count	get data cycle size counter
• ddp_queue_clear_cycle_count	clear data cycle size counter
• ddp_get_enqueue_buf	get write buffer from queue

Socket Private Library Object

The socket library contains socket management utilities. The TCP socket routines are responsible for the server and clients communications.

- ddp_accept_socket — accept client connection
- ddp_close_socket — close client connection
- ddp_set_connect_timeout — enable/disable socket connect timeout
- ddp_create_socket — initiate connection
- ddp_open_socket — connect client to server
- ddp_shutdown_socket_no_send — close outgoing halp of a socket
- ddp_write_from_queue — write into socket from queue
- ddp_read_from_queue — read into queue from socket
- ddp_send_accept — send accept packet
- ddp_send_connect — send connect packet
- ddp_send_control — send control packet
- ddp_send_cycle — send cycle packet
- ddp_send_disconnect — send disconnect packet
- ddp_send_enable — send enable packet
- ddp_send_publish — send publish packet
- ddp_send_response — send response packet
- ddp_send_start — send start packet
- ddp_send_stop — send stop packet
- ddp_send_subscribe — send subscribe packet
- ddp_send_unsubscribe — send unsubscribe packet
- ddp_send_unpublish — send unpublish packet
- ddp_send_value — send value packet
- ddp_alloc_circular_buffer — allocate a circular buffer
- ddp_write_buffer — write the buffer out

APPENDIX A

Statement of Work

- Provide performance data for system modeling.
- Confirm and or modify system data flow for FD Data Distribution.
- Provide the capability for the Data Distribution function to be utilized in both Operational and Application configurations.

DDP Data Merger Function

- Collect Gateway Change Data packets from all gateways at the system synchronous rate.
- Collect Application Change Data packets from all CCPs at System synchronous rate (*No Application Change Data packets until Thor Delivery*).
- Merge Gateway Change Data and Application Change Data in to a single a stream ordered to the nearest 0.1 ms. (*No Application Change Data until Thor Delivery*).
- Merge health data into data element from health table.
- Place requested FDs in queues for the Data Fusion Function.
- Place requested FDs in queues for the Data Health Function.
- Place requested FDs in queues for the Data Constraint Function (*Data Constraint Function is not part of Redstone Delivery*).
- Transmit this data at system synchronous rate on the RTCN.
- Transmit this data at display synchronous rate on the DCN.
- Define and provide a method to send System Default Display Data Attribute Values. (*A placeholder will be reserved in the CVT for default Display Data Attribute values. Setting mechanism will be defined for the Thor delivery*).
- Maintain statistics on packet rates, data rates, and CPU utilization.

CCP Data Function

- Collect RTCN Change Data Packets from the DDP at system synchronous rate.
- Place requested FDs in queues for System and User Application.
- Provide an output queue for user Application Derived FDs and transmits them to the DDP at system synchronous rate. (*No user Application Derived FDs to transmit in Redstone*).
- Maintain statistics on packet rates, data rates, and CPU utilization.

HCI Data Function

- Collect DCN Change Data Packets from the DDP at display synchronous rate.
- Place requested FDs in queues for System and User Application.
- Maintain statistics on packet rates, data rates, and CPU utilization.

Current Value Table Function

- Maintain in the DDPs, CCPs and HCIs a Current Value Table that contains for all FDs the current data value, its health and time of last change.
- Support all FD type including Time Homogenous and Multiword data.
- Provide separation of data for different flow zones. (Added during kickoff meeting)